CSE 331 18au Section 1 – Specifications

1. **Alice must write a method histogram that takes in an array of integers sleepData that corresponds to answers from a survey about how many hours college students sleep, with possible answers ranging from 0 to 9. histogram then returns an array of integers of size 10, where the value at position \( i \) is the number of times \( i \) appeared in sleepData. For example, if sleepData = [3, 4, 6, 7, 2, 1, 4], then histogram returns [0, 1, 1, 1, 2, 0, 1, 1, 0, 0]. If sleepData is null, throw a NullPointerException, and if sleepData is empty, return null. Fill out histogram’s specification:**

```java
/**
 * @spec.requires __________________________________________________________
 * @spec.modifies __________________________________________________________
 * @spec.effects __________________________________________________________
 * @return _______________________________________________________________
 * @throws _________________________ _______________________________________
 */
public int[] histogram (int[] sleepData) {
```

2. **Implement the following specification:**

```java
/** Given two integer side lengths \( a \) and \( b \) of a triangle, returns the largest possible integer value of the third side \( c \)
 * @spec.requires \( a > 0, b > 0 \)
 * @returns the largest possible integer value of \( c \).
 * @throws NullPointerException if \( a == null \) or \( b == null \)
 **/
public int largestSideLength (int a, int b) {
```
3. *Suppose we have a* BankAccount *class with instance variable* balance. *Consider the following three specifications for a* BankAccount *method* withdraw, *which takes in an int amount that signifies the amount the user wants withdrawn from the balance:*

A. *@spec.effects decreases* balance *by amount.*

B. *@spec.requires* amount $\geq 0$ and amount $\leq$ balance *@spec.effects decreases* balance *by amount.*

C. *@spec.effects decreases* balance *by amount*
   *@throws* InsufficientFundsException *if* balance $<$ amount

*Which specifications do each of these implementations meet? Write A, B, and/or C for each implementation.*

I. ```java
void withdraw(int amount) {
    balance -= amount;
}
```
   *Specifications: _________________________________

II. ```java
void withdraw(int amount) {
    if (balance $\geq$ amount) {
        balance -= amount;
    }
}
```
   *Specifications: _________________________________

III. ```java
void withdraw(int amount) {
    if (amount $<$ 0) {
        throw new IllegalArgumentException();
    }
    balance -= amount;
}
```
   *Specifications: _________________________________

IV. ```java
void withdraw(int amount) throws InsufficientFundsException {
    if (balance $<$ amount) {
        throw new InsufficientFundsException();
    }
    balance -= amount;
}
```
   *Specifications: _________________________________
4. (Midterm 15wi Problem 4) Here is the header for a method that computes a student’s overall score and adds that information to a gradebook data structure:
void addScore(String name, List scores, Map gradeBook);

A. Here are two possible specifications for this method:

X
@spec.requires name != null and scores != null and gradeBook != null
@spec.modifies gradebook
@spec.effects add a mapping to gradebook

Y
@spec.requires name != null and scores != null
@spec.modifies gradebook
@spec.effects add a mapping to gradebook
@throws IllegalArgumentException if gradeBook is null

Which specification is stronger than the other? (circle)  X  Y  neither

B. Here is one possible implementation of this method:
if (name == null || scores == null || gradeBook == null)
{
    throw new IllegalArgumentException();
}
double grade = 0.0;
for (double s : scores) {
    grade += s;
}
if (scores.size() > 0) {
    grade /= scores.size();
}
gradeBook.put(name, grade);

Which specification(s) does this implementation satisfy? (circle)  X  Y  both  neither
5. (Midterm 17AU Problem 1) Alice is writing a function $\texttt{bestDeal}$ that takes in an array and then returns the smallest price. She intends to implement $\texttt{bestDeal}$ by sorting prices, but she does not want clients to depend on prices being sorted.

A. Write a specification for her function:

```c
/**
 * *
 * *
 * *
 * *
 * *
 * /
int bestDeal(int[] prices) { ... }
```

B. Suppose that Alice decides to change her implementation to no longer sort prices. How should she change the specification above?

C. This new specification would be (circle one): weaker incomparable stronger

D. Suppose that Alice decides instead to stick with the version that sorts prices but will now allow clients to depend on that behavior. How should she change the specification above?

E. This new specification would be (circle one): weaker incomparable stronger